Chapter 18

Derandomization using Conditional Expectations

Yes, my guard stood hard when abstract threats
Too noble to neglect
Deceived me into thinking
I had something to protect
Good and bad, I define these terms
Quite clear, no doubt, somehow
Ah, but I was so much older then
I'm younger than that now

My Back Pages, Bob Dylan

By Sariel Har-Peled, March 19, 2024[1]

18.1. Method of conditional expectations

Imagine that we have a randomized algorithm that uses as randomized input n bits $X_1, \ldots, X_n$, and outputs a solution of quality $f(X_1, \ldots, X_n)$. Assume that given values $v_1, \ldots, v_k \in \{0, 1\}$, one can compute, efficiently and deterministicly, the quantity

$$\mathbb{E} f(v_1, \ldots, v_k) = \mathbb{E}[f(v_1, \ldots, v_k, X_{k+1}, \ldots, X_n)] = \mathbb{E}[f(X_1, \ldots, X_n) \mid X_1 = v_1, \ldots, X_k = v_k]$$

by a given procedure **eval**$_{\mathbb{E}f}$. In such settings, one can compute efficiently and deterministicly an assignment $v_1, \ldots, v_n$, such that

$$f(v_1, \ldots, v_n) \geq \mathbb{E} f, \qquad \text{where} \qquad \mathbb{E} f = \mathbb{E}[f(X_1, \ldots, X_n)].$$

Or alternatively, one can find an assignment $u_1, \ldots, u_n$ such that $f(u_1, \ldots, u_n) \leq \mathbb{E}[f(X_1, \ldots, X_n)]$.

The algorithm. Assume the algorithm had computed a partial assignment for $v_1, \ldots, v_k$, such that $\alpha_k = \mathbb{E} f(v_1, \ldots, v_k) \geq \mathbb{E} f$. The algorithm then would compute the two values

$$\alpha_{k,0} = \mathbb{E} f(v_1, \ldots, v_k, 0) \qquad \text{and} \qquad \alpha_{k,1} = \mathbb{E} f(v_1, \ldots, v_k, 1).$$

Observe that

$$\alpha_k = \mathbb{E} f(v_1, \ldots, v_k) = \mathbb{P}[X_{k+1} = 0] \mathbb{E} f(v_1, \ldots, v_k, 0) + \mathbb{P}[X_{k+1} = 1] \mathbb{E} f(v_1, \ldots, v_k, 1) = \frac{\alpha_{k,0} + \alpha_{k,1}}{2}.$$

As such, there is an i, such that $\alpha_{k,i} \geq \alpha_k$. The algorithm sets $v_{k+1} = i$, and continues to the next iteration.

Correctness. This is hopefully clear. Initially, $\alpha_0 = \mathbb{E} f$. In each iteration, the algorithm makes a choice, such that $\alpha_k \geq \alpha_{k-1}$. Thus,

$$\alpha_n = \mathbb{E} f(v_1, \ldots, v_n) = f(v_1, \ldots, v_n) \geq \alpha_{n-1} \geq \cdots \geq \alpha_0 = \mathbb{E} f.$$

Running time. The algorithm performs $2n$ invocations of **eval**$_{\mathbb{E}f}$.

Result.

Theorem 18.1.1. *Given a function $f(X_1, \ldots, X_n)$ over n random binary variables, such that one can compute determinedly $\mathbb{E}f(v_1, \ldots v_k) = \mathbb{E}[f(X_1, \ldots, X_n) \mid X_1 = v_1, \ldots, X_k = v_k]$ in $T(n)$ time. Then, one can compute an assignment $v_1, \ldots, v_n$, such that $f(v_1, \ldots, v_n) \geq \mathbb{E}f = \mathbb{E}[f(X_1, \ldots, X_n)]$. The running time of the algorithm is $O(n + nT(n))$.*

18.1.1. Applications

18.1.1.1. Max kSAT

Given a boolean formula F with n variables and m clauses, where each clause has exactly k literals, let $f(X_1, \ldots, X_n)$ be the number of clauses the assignment $X_1, \ldots, X_n$ satisfies. Clearly, one can compute f in $O(mk)$ time. More generally, given a partial assignment $v_1, \ldots, v_k$, one can compute $\alpha_k = \mathbb{E}f(v_1, \ldots, v_k)$. Indeed, scan F and assign all the literals that depends on the variables $X_1, \ldots, X_k$ their values. A literal evaluating to one satisfies its clause, and we count it as such. What remains are clauses with at most k literals. A literal with i literals, have probability *exactly* $1 - 1/2^i$ to be satisfied. Thus, summing these probabilities on these leftover clauses given use the desired value. This takes $O(mk)$ time. Using Theorem 18.1.1 we get the following.

Lemma 18.1.2. *Let F be a kSAT formula with n variables and m clauses. One can compute deterministicly an assignment that satisfies at least $(1 - 1/2^k)m$ clauses of F. This takes $O(mnk)$ time.*

18.1.1.2. Max cut

18.1.1.3. Turán theorem

Lemma 18.1.3 (Turán's theorem). *Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with n vertices and m edges. One can compute determinedly, in $O(nm)$ time, an independent set of size at least $\dfrac{n}{1 + 2m/n}$.*

Proof: Exercise. ■

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.